

Klausur zur Veranstaltung  
Softwareentwicklung 1  
Sommersemester 2004

Hessische VWA

Dr. Alexandra Roder, Bernd Ulmann

5. Juni 2004

Hinweise:

- Die Klausur besteht aus 24 Teilaufgaben.
- Insgesamt sind 120 Punkte erreichbar, jede Teilaufgabe entspricht hierbei 5 Punkten.
- Als Hilfsmittel sind alle „unbelebten Hilfsmittel“, d.h. insbesondere Ihre Vorlesungsmitschriften zugelassen.
- Bitte schreiben Sie leserlich! (Dankeschön! :-)
- Verwenden Sie **keinen** Bleistift!

Viel Erfolg!

## Aufgabe 1

Sie sind Leiter eines Projektes für eine Dialogfunktion, das voraussichtlich einen Umfang von 20000 Zeilen Code aufweisen wird. Zu dessen Durchführung stehen Ihnen drei Programmierer zur Verfügung, die Monatsleistungen von 600, 300 und 300 Zeilen erbringen.

1. Berechnen Sie den Aufwand in Personenmonaten, wenn man von einer durchschnittlichen Programmierleistung ausgeht.

2. Berechnen Sie die optimale Laufzeit des Projektes.

3. Berechnen Sie die benötigte Anzahl von Programmierern, um das Projekt so schnell wie möglich zum Abschluß zu bringen, wenn die durchschnittliche Programmierleistung gleich bleibt.

4. Die folgende Aufgabe soll den Vorteil, den Programmierer gegenüber anderen Tierarten besitzen, aufzeigen. Als Beispiel hierfür diene eine Normrennechse, die mit einer Maximalgeschwindigkeit von  $29 \frac{km}{h}$  läuft (was sie im weiteren auch praktisch tut).

Wie schnell laufen zwei solcher Normrennechsen, die mit maximaler Geschwindigkeit laufen?



3. Zeichnen Sie nun das vollständige Diagramm des Zustandsautomaten, aufbauend auf den zuvor identifizierten sechs Hauptzuständen.

## Aufgabe 3

Die folgende Aufgabe bezieht sich auf die Erstellung eines Klassendiagramms im Rahmen der Verwaltung eines Flugbetriebes. Zu betrachten sind hierbei:

- Flüge
- Flugpersonal
- Kunden

Je Flug sind ein Pilot, ein Copilot sowie drei Flugbegleiter vorzusehen – wichtig ist, daß jeder Pilot der Vorgesetzte des ihm zugeordneten Copiloten ist und darüberhinaus Copiloten weniger als Piloten (aber mehr als Busfahrer) verdienen. Das Flugpersonal verfügt über die Attribute *Gehalt*, *Name* sowie eine *Personalnummer*. Von den Kunden sollen *Name*, *Adresse* und *Kundennummer* gespeichert werden.

Die Flüge zeichnen sich durch *Flugnummer*, *Ankunftszeit*, *Abflugzeit* sowie die Art des Fluges aus (*Langstrecke* beziehungsweise *Kurzstrecke*). Weiterhin müssen *Abflug-* und *Ankunftsort* berücksichtigt werden.

1. Zeichnen Sie die Klassen **Kunde** und **Personal**. Hierbei ist eine Möglichkeit zur Abfrage des Gehalts eines Angehörigen des Flugpersonals vorzusehen!

2. Zeichnen Sie die Klasse **Flug**, wobei die Attributtypen mitanzugeben sind und zu berücksichtigen ist, daß die Art des Fluges nur die Ausprägungen **Kurzstrecke** beziehungsweise **Langstrecke** annehmen darf.

**3. Die folgende Aufgabe zählt 12 Punkte!**

Zeichnen Sie das Klassendiagramm mit allen notwendigen Assoziationen und Restriktionen (und bedenken Sie, daß eine selbstbezügliche Assoziation auftritt!).

## Aufgabe 4

Die folgende Aufgabe behandelte einen kleinen Ausschnitt des EDV-Systems eines Bankhauses. Darin verwalten Banksachbearbeiter Kundendaten, während Kunden beim Geldabheben auf Kontoinformationen zugreifen. Weiterhin ist für den Vorgang des Geldabhebens ein Zugriff auf die Kundendaten notwendig – für diesen Ausschnitt soll letztlich ein Datenflußdiagramm erstellt werden.

1. Identifizieren und benennen Sie die nötigen Prozesse.
2. Geben Sie die nötigen Schnittstellen und Datenspeicher an.
3. Identifizieren und benennen Sie die nötigen Datenflüsse.

4. Zeichnen Sie das Datenflußdiagramm.

## Aufgabe 5

Die folgende Aufgabe widmet sich einer Variation der Ihnen sicherlich hinreichend bekannten Fibonacci-Folge, die im Gegensatz zu dieser darauf beruht, daß jedes Folgenglied die Summe seiner *drei* direkten Vorgänger ist, es gilt also

$$\begin{aligned}a_0 &= \\a_1 &= \\a_2 &= 1 \quad \text{und} \\a_i &= a_{i-1} + a_{i-2} + a_{i-3},\end{aligned}$$

wobei die einzelnen Folgenglieder mit  $a_i$  bezeichnet werden.

1. Geben Sie die **Werte** der Folgenglieder  $a_4$  bis  $a_9$  an.

2. Zur Berechnung der  $a_i$  werde folgende, in C formulierte Funktion verwendet:

```
unsigned int a_rek (unsigned int i)
{
    if (i < 3)
        return 1;

    return a_rek (i - 1) + a_ref (i - 2) + a_ref (i - 3);
}
```

Zeichnen Sie einen Baum, welcher die im Zuge der Berechnung von `a_rek (5)` auftretenden, verschachtelten Funktionsaufrufe wiedergibt (analog zu den Beispielen aus der Vorlesung, welche dies anhand der Fibonacci-Folge erläuterten).

3. Wieviele Funktionsaufrufe der rekursiven Funktion `a_rek ()` sind für die Berechnung von `a_ (5)` notwendig (durch Abzählen an Ihrem Aufrufbaum aus der vorangegangenen Aufgabe – der initiale Aufruf von `a_rek (5)` ist übrigens mitzuzählen!)?
  
4. Als nächste Frage ist nun zu klären, wieviele Funktionsaufrufe der Aufruf von `a_rek (6)` benötigt (keinen Baum zeichnen – zumindest nicht in das Klausurheft :-)).
  
5. Und nun die Gretchenfrage: Wie stark wächst die Laufzeit der Funktion `a_rek (i)` mit steigendem `i` an? Sicherlich mehr als linear! Aber wie stark wirklich (Sie dürfen natürlich, wie stets in solchen Fällen, stark vereinfachen!)? Quadratisch? Oder eher kubisch? Oder irgendetwas anderes? Halten Sie sich vor Augen, was geschieht, wenn `i` um eins vergrößert wird.

6. Nachdem nun klar ist, daß die rekursive Berechnung von  $a_i$  nicht wirklich performant ist, liegt folgende iterative Variante nahe:

```
#include <stdio.h>

unsigned int a_iterativ (unsigned int i)
{
    int a_0, a_1, a_2, a_i, j;

    if (i < 3)
        return 1;

    for (j = a_0 = a_1 = a_2 = 1; j < i - 1; j++)
    {
        a_i = a_0 + a_1 + a_2;
        a_0 = a_1;
        a_1 = a_2;
        a_2 = a_i;
    }

    return a_i;
}

int main ()
{
    printf ("a(5)=%d, a(6)=%d\n", a_iterativ(5), a_iterativ(6));
    return 0;
}
```

Wieviele Schleifendurchläufe benötigt hierbei die Berechnung von `a_iterativ(5)`?

7. Und wieviele Schleifendurchläufe benötigt die Berechnung von `a_iterativ(6)`?

8. Und nun: Wie verhält sich die Laufzeit der Routine `a_iterativ(i)` für wachsendes  $i$ ? Wächst sie linear? Quadratisch? Kubisch? Irgendwie anders?

## Aufgabe 6

Zum Abschluß das fast obligatorische Minirätsel:

```
#include <stdio.h>

int i;

int murks (int wert)
{
    int j, summe;

    if (wert > i)
i = i + wert;

    summe = 0;
    for (j = 1; j <= i; j++)
summe = summe + j;

    return summe;
}

int main ()
{
    int j;

    i = 3;
    j = murks (7);

    printf ("j=%d, i=%d\n", j, i);
    return 0;
}
```

1. Welchen Wert besitzt j zum Zeitpunkt seiner Ausgabe?

2. Und welchen Wert besitzt i zum Zeitpunkt seiner Ausgabe?